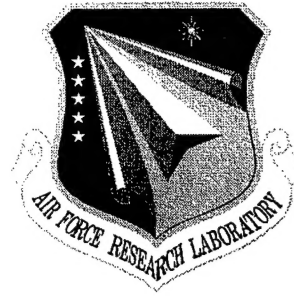


AFRL-IF-RS-TR-2000-167
Final Technical Report
December 2000



INTEGRATING C4ISR MODELS ACROSS DIFFERENT LEVELS OF ABSTRACTION

PDHP Technology

Thomas C. Fall

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK**

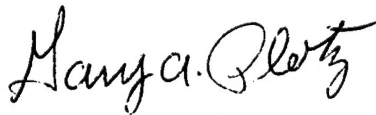
DTIC QUALITY INSPECTED 1

20010220 037

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

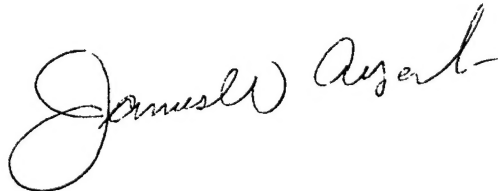
AFRL-IF-RS-TR-2000-167 has been reviewed and is approved for publication.

APPROVED:



GARY A. PLOTZ
Project Engineer

FOR THE DIRECTOR:



JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFSB, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE DECEMBER 2000		3. REPORT TYPE AND DATES COVERED Final Mar 99 - Jun 00
4. TITLE AND SUBTITLE INTEGRATING C4ISR MODELS ACROSS DIFFERENT LEVELS OF ABSTRACTION			5. FUNDING NUMBERS C - F30602-98-C-0288 PE - 62702F PR - 459S TA - BA WU - 81	
6. AUTHOR(S) Thomas C. Fall				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) PDHP Technology P.O. Box 1365 Laramie WY 8203			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFSB 525 Brooks Road Rome NY 13441-4505			10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2000-167	
11. SUPPLEMENTARY NOTES Air Force Research Laboratory Project Engineer: Gary A. Plotz/IFSB/(315) 330-4383				
12a. DISTRIBUTION AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) Modeling and simulation has proven an effective tool for analyzing C4ISR systems. C4ISR systems that have been modeled to date include theater, mission, engagement and engineering level systems. However, these models address the C4ISR issues at different levels of detail in time-step, number, and range of variables. Each of these models operates at a relatively fixed level of detail within a well understood framework and is, therefore, relatively easy to simulate and analyze. This means that not just the model, but also the interacting effects of model variables using different levels of detail must be analyzed. To analyze these systems requires more complex tools that can analyze how the choice of value for the stochastic variable, passed between the differing models, impacts the overall simulation behavior. Without this capability, current simulations can only analyze a limited range of possible model results. Data choices that might have a critical impact on model behavior could be missed. The Dynamic Focusing Architecture (DFA) approach is one solution to this problem.				
14. SUBJECT TERMS Modeling and Simulation, Model Abstraction			15. NUMBER OF PAGES 48	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Table of Contents

1.0 Introduction	1
2.0 An overview of the DFA approach	3
3.0 Examples of use of DFA	9
3.1 Use of DFA to analyze systems of systems	9
3.2 Use of DFA to analyze needs for a military SATCOM system	22
4.0 Issues concerning Metamodeling	30
4.1 What are metamodels?	30
4.2 How would metamodels work	31
4.3 Why metamodeling isn't easy	33
4.4 How can we accomplish the metamodel	34
5.0 Issues concerning the DFA tool	36
6.0 Summary	37

List of Figures

Figure 2.1:	Two Different Trajectories	6
Figure 3.2:	Simple linear approximation compared to 'real'	18
Figure 3.3:	A better single valued approximation to the distribution	19
Figure 3.4:	Approximating the distribution using the average high, average low	20
Figure 3.5:	Distribution after consideration of other parameters	21
Figure 3.6:	Flow Chart One Combat Turn	24
Figure 3.7:	A screen dump of our model	25

List of Tables

Table 3.1:	Initial values for the four runs	16
Table 3.2:	Results of the 4 runs over three runs	16

1.0 Introduction

Modeling and simulation has proven itself as a tool for many aspects of the latter portions of a weapon system's development and deployment life cycle, from engineering trades to cost models to training simulators. However, the very early phases of weapons still use the 'back of the envelope' as the primary development tool. And this will always probably be the tool of choice, since that is how the few individuals who carry the reservoirs of corporate knowledge work together. Up to now, those envelopes could be transferred directly to the other stage of the development process because we were in a world where the overall requirements were quite well delineated. It was a world where there were strategic forces on each side that needed to be balanced and tactical forces that needed to be balanced. As such, future weapon systems could be judged against a well-understood framework. Now, we are in a world that does not have this single overarching framework. This makes life a little harder at the 'back of the envelope' level, but now we have real issues moving these ideas down to the development troops. The high level people were always comfortable with degrees of uncertainty, but the Cold War framework allowed the transfer process to restructure to a highly delineated frame. Without this frame, we have a gap in our development process. The Dynamic Focusing Architecture (DFA) is aimed at infrastructuring the solution to that gap.

The 'back of the envelope' would depict the important elements, their nominal values, the computational relation between them and finally, the implications of these: "If we can achieve these values for these elements, then it will provide this much benefit (with the unstated 'under the assumption that the environment elements have these values')". In the Cold War days, the whole community pretty well shared an idea of what these environment values were. Our situation now is that we must have a layer of analysis that helps us delineate which environment variables are important and for those, in what ranges does solution work as opposed to another

solution working in other ranges. Later in this paper we will discuss a nominal example of this, namely how the degree of dispersal of the threat forces can affect choice of weapon systems, and a real world example, namely how assumptions about communications traffic can affect the needs for capabilities on a SATCOM payload.

Let's look at how this process would work. These decision situations have some elements of high uncertainty as well as some elements of low uncertainty, both for the system and for its environment. It may be that in actuality, only a few of those elements are really of importance. What we would like to do is to identify those so that we can concentrate our analysis resources on them, postponing the less salient elements till their analysis becomes important. The Dynamic Focusing Architecture is designed precisely for this task. We can see whether there are components of the environment that produce large amounts of variability in the benefit of the system. If we can identify these, then we can decide whether to try to make one system that addresses the whole range or different systems to address particular types of environments.

With this in mind, this report will try to show how DFA could be used. We will start off with an overview of the DFA approach, and then descriptions of two analyses that have been done using DFA, one a nominal Systems of Systems study, the other an analysis of utility to help define a military SATCOM's user's needs. These efforts give a springboard for how we can attack the larger problems that modeling and simulation needs to address, that is how does it assist in the acquisition of systems with long development cycles being used for rapidly changing, ill understood future crisis and conflict situations. We will then discuss where DFA approach is now as a metamodel and as a tool: what it could be immediately useful for and what needs to be done to make it more widely useful in these longer range analysis efforts.

2.0 An overview of the DFA approach

The dynamic focusing approach (DFA) addresses the issue of how to model a dynamic system where some regions of its modeling space would require more attention than others do. DFA seeks to identify regions that evolve divergently, especially those evolving into interesting spaces, as opposed to those that evolve convergently. For the latter, one replication at a coarse level can pretty well define the evolution that would be applicable to any portion of the region. In the former case, we would need to divide the regions and model each of those. Some of these subregions might have convergent evolution, others would diverge. The divergent regions, if they seemed to evolve into interesting places, could be subdivided even further. A point should be clarified here. We are looking at a modeling space, that is, a space of possible initial conditions, boundary conditions and states and we are examining the ways a particular assignment of values to the initial and boundary parameters will evolve. That is, a trajectory is the evolutionary path of the source system under these conditions and then we have the simulation that is the computational model that tries to achieve this same trajectory.

If the source system has some regions of chaos, then if the model is a good representation, it will show divergent trajectories when run on condition values that are approximately in this region. If some of these simulations evolve into states that are particularly interesting (e.g., in combat simulations, the side that under most conditions had been losing is able to hold its own under these), we would like to identify that and to examine it further. An analogy might be hydraulic flow, say a wind tunnel, where there are regions of turbulent flow and regions of planar flow. What DFA does is similar to injecting smoke marker streams in the wind tunnel, which show us the planar flow regions where the marker streams move in parallel, and the turbulent regions, where some marker streams might curl into vortices, but at a minimum the markers strongly diverge. In the chaotic regions, points that start close together can end up very far apart. Though

we might not be able to predict exactly where the portions will end up, the presence of the marker stream at least alerts us to the fact that the region is highly sensitive. Following this hydraulic flow analogy, we want to follow enough trajectories to identify a) when we get divergence and b) when trajectories evolve into regions of interest. If we could characterize these conditions as distributions, then in a sense we are trying to propagate distributions. The problem is that some of these distributions depend on the states of objects within the simulation, which can evolve differently. For this reason, we would really like to have a methodology that lets us look at individual trajectories, but in a smart way.

DFA is a tool that allows us to trace trajectories in a smart way for systems that are decomposable. To explicate its methodology, we'll compare it to other modeling techniques. According to Ziegler, we can divide models into three categories: differential equations, difference equations, and discrete event simulators. The first category, differential equations, is primarily used in cases where we can do analytic solutions. They are a powerful expressive form, but because they are typically hard to analytically solve, we often used an augmented form of the second category, time difference equations, to obtain solutions. So when we talk about simulations, we usually confine ourselves to the latter two categories. The difference equation category is usually seen as a 'time stepped' simulation. All of the elements move from their states at one time click to their states at the next time click. Cellular automata, such as Conway's Life, are an example of this category that might not necessarily immediately be seen as difference equations, but are very powerful concepts. Cellular automata are a set of uniform elements connected into a uniform topology. Each cell has the same set of potential states and the same connection pattern to neighboring elements. Each has the same set of rules that determine the element's next state based on its current state and the states of its neighbors. If the rules have at least a fairly low degree of complexity, the cellular automata they define will be a universal computing device. The Turing machine could be considered a one-dimensional

cellular automaton. For these devices above that threshold, one can trade complexity of rules against complexity of initial conditions.

As powerful as these time-stepped simulators might be, they have two problems: 1) they are inefficient; every cell must be examined even if there is no possibility that it will change state, and 2) the simulator typically has a smallest time scale, its underlying click size. The third category, discrete event simulators, addresses these two problems; an element is only examined if and when there is an event occurring to it that would change its state. Thus, elements that are not changing state are not examined, which addresses the efficiency issue, and the events take place when they are scheduled, there is no *a priori* time click size. If we want finer time resolution, we include elements that react at the finer time level.

For a single replication, discrete event simulation is unparalleled. Our problem arises when we want to explore a modeling space where each of the elements can take a range of states. In this case, we must make several replications to capture the different ways the states of the elements can interact. One problem is that slightly different states can cause slightly different timings. For the computer, it is hard to recognize that substantial changes of order of events can actually represent trajectories that are fairly close if these are all separated by a fairly short time interval. We will look into an example showing this shortly.

Often, a discrete event simulator, to explore a space of possibilities, must use the Monte Carlo approach. That is, stochastic choices are made at various points to reflect the range of possibility for that variable. Each replication represents a trajectory for the system's evolution. This brings us back to the problem of combining distributions. Even though it is called Monte Carlo technique, it was not originally designed for stochastic purposes; rather, in the fifties, it was designed as a technique to do numerical integration. The choice of values for each independent variable is similar to the choice of initial condition for our evolutions. It comes down to, what is

the best strategy to pick values that explore our problem space. In the Monte Carlo approach, each variable is given a set of values representing the range that variable can take as well as a distribution for that range, which represents how often each of those values should be output. Where there are dependencies between variables, some expression of their relation must be provided. In the discrete event simulator, this process is effectuated by events. When an event occurs to an element, it brings with it a call to one of that element's methods and values for some of the variables that method requires. The remainder of the required variables would be resident in the element as state values. The method would be run with these values and would produce new values for the element's states and new events which individually consist of a time, an element identifier, a method identifier and variable values. If we were exploring a problem space, any of these event parameters could be stochastically chosen.

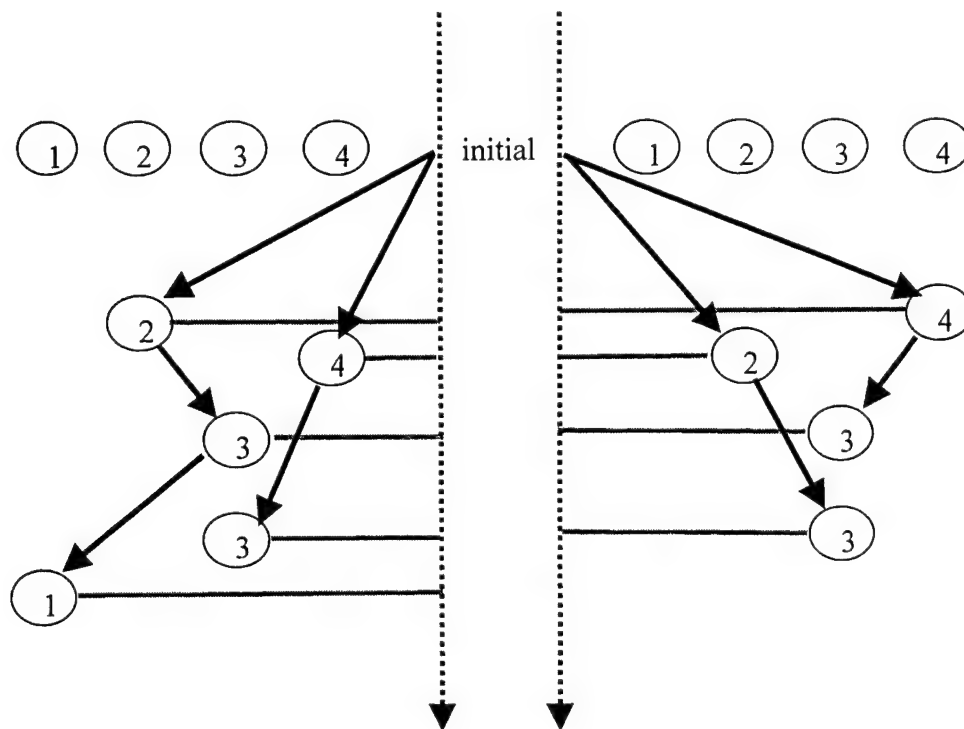


Figure 2.1: Two different trajectories

Figure 2.1, "Two different trajectories", shows an issue that can arise when we are tracing multiple trajectories. In both cases, at initialization we schedule two events, one for element 2 and one for element 4. On the left, the event for element 2 is scheduled first and it schedules an event for element 3, which in turn schedules an event for element 1. The event at element 4 also causes an event to be scheduled for element 3. On the right, the events for elements 2 and 4 also cause events to be scheduled for element 3, but now the event from 4 precedes the event from 2. The event from 2 sees element 3 in a different state than it had seen in the left hand case and now there is no event scheduled for element 1. Small differences at these individual decision points could cause major differences in the total trajectories. The Monte Carlo method, by exercising different appropriate choices, explores different trajectories on each replication. However, since most distributions are normally distributed, the typical path will be fairly close to the average path. From the early days, it was recognized that by itself, Monte Carlo would require a large number of replications to identify behavior in the tails. Techniques such as weighted Monte Carlo were developed, where the distributions were partitioned and the point at the center of gravity was used to represent the entire partition. The Monte Carlo chose from those points and carried the weight along. Clearly, if the partitions are done so that for each partition, each has the same mass, much less bookkeeping is needed.

This weighted technique converges much more rapidly, but still, for any set of replications, since the choices are random, you do not know for sure whether every possible path has been tried and some paths may have been followed more than once. One way is to control the choices using some strategy. Doing this has one major advantage: if we look at one particular result, we know how it got there. In particular, if there was some region of the output space that we were particularly interested in, we could trace back to see if there were any decisions that consistently put us into that region. The most robust strategy that will not miss any interesting path is to

enumerate all the possible paths and to follow each one. This has the disadvantage, of course, that it is exponentially explosive.

The DFA currently implemented uses a controlled Monte Carlo search; in fact it uses the exhaustive approach. Each distribution is divided into two equal partitions, the portion above the median and the portion below the median and the center of mass of each is determined (we call these the *average low* and the *average high*). So, if there are n different variables, the number of paths goes up as 2^n . If the computations for 10 variables were a millisecond, then for 20 it would take a second, but 30 would take 17 minutes and 40 would take 12 days. Clearly, we would like to keep the number of variables no worse than the mid-twenties. This is why earlier we had said DFA is applicable to decomposable systems, that is systems where we can decompose them into subsystems, each with a sufficiently small number of variables. Each subsystem is run, its resulting variables are aggregated into average low and average high, and these are passed to a computation where they are combined to compute the total system. A system composed of 25 subsystems each with 25 variables would take less than half an hour.

In actual usage, we stand this bottoms-up approach on its head and go from the top down. That is, we have default values for the subsystem supplied variables that are used by the system level computation. If there are regions where its outputs look interesting, we can use our audit trail to determine which choices for which subsystems were made that got us there. If we find that there are a couple of subsystems where we consistently made the same choice, then we could look at those in more detail. These are initialized appropriately for the scenario and are run. The newly computed values are used to replace the default values and we rerun the system computations. At this point, we might look at other subsystems, or we could decompose one of the subsystems we'd just run to the subsystem. Thus, we can dynamically focus our computational and analytic resources into those areas providing the most information for our problem.

3.0 Examples of use of DFA

During the course of this effort, two important features were added to the DFA engine, time stepping and branching. The initial version would take a set of variables defined with their average low and average high and a computational definition using these variables and produce an output distribution, from which we could get an average high and average low. However, if we are looking at a dynamic system, there are usually several variables that modify each other during one time step and initialize the next. For the time stepping capability, we collect each of these and calculate their average high and low and pass that to the next time step. To collect the average high and average low of each variable, we must individually sort it and find the median. This increases the memory management problems, but provides the ability to follow dynamic systems stochastically, an informative journey. As mentioned, the usual time stepping mechanisms follow only a single computational thread, missing the very important information about sensitivity, e.g., whether we are in a convergent trajectory or a divergent. In the initial DFA, we had all the computation mechanisms, but when moving to the infrastructure for the time stepping, the branching capability had to be revisited. Branching, combined with the time stepping, provides a very powerful tool. Combat situations require that available resources be apportioned to deal with opposition force elements. Branching allows us to do this on a thread by thread basis. One choice of values for the variables might lead us to a situation where we go down one branch, whereas other choices of values for initial variables might lead us down other branches.

3.1 Use of DFA to analyze Systems of Systems

Though recent economic history seems to indicate that focussing on a narrow niche, that is confining attention to a single system, is the path to success and riches. However, there are

many undertakings that cannot be measured only by themselves. Rather, they must be measured by how they enhance the value of an overarching system. DOD in particular is faced with this problem. Weapon systems are procured by individual services and even within a service, each system's acquisition is pretty much standalone. And yet, the value of any individual system is really in how it works with the other fielded systems in a combat effort. It may more than meet its required performance, but if that performance is not needed, then the system is not adding value. This depends on the opposition forces systems and objectives. One of the issues being faced now is that if the opposition has diffuse objectives, counter-objectives cannot be well focussed. One of the problems facing analysts is that not only is there uncertainty in what the opposition might have as resources and objectives, but there is also uncertainty in what our weapons system's effectiveness might be. This uncertainty has many sources: besides the unknown opposition, there can be factors such as their disposition. For example, bin Laden's forces in Afghanistan were small and relatively lightly armed, but their disposition in tunnels into steep mountainsides made it extremely difficult for cruise missiles to have much effect. To gauge the value of a system, somehow we must cover this breadth of possibilities. This is what the Dynamic Focusing Architecture (DFA) was designed for.

We will give a very high level example of a model of this situation that is implemented using DFA, discuss how one analyzes the results of the runs, then discuss how one proceeds to richer analyses. One of the issues is how do we sensibly make our high level cuts so that the simulated world behaves, at this grosser level, as much as possible like we would expect the real world to behave. Our goal at this model design level is to divide the behavior space into independent portions that can then be characterized by a single parameter. The military board games provide a convenient analog. A typical game will concern a particular type of warfare, e.g., tank combat, at a particular level, say battalion. Game pieces, representing the smallest elements the game plays, and a terrain grid whose cells are an appropriate size for those smallest units, are provided. For a given game, there is an order of battle (OOB) for each side as well as positional

information. Finally, there is an objective for each side. Each player draws the game pieces appropriate for his OOB and, places them on the board within the constraints of his boundaries. The game now proceeds by turns in which the players move and then engage opposition elements on cells adjoining cells of their combat elements. Each game piece has information as to what type of unit it is, its mobility factor and its combat strength. The mobility factor is used during the movement part of the turn and the combat strength is used during the engagement portion. Adding all of the defending elements and all of the attacking elements and getting the ratio indicate the engagement outcomes. This ratio brings us to a certain row of the game's combat results table that has several outcomes depending on a stochastic draw. For manual games, a die is used and the combat results table will have six outcomes. The die is rolled and from the table's entry we get the results of that engagement, namely a decrease in combat strength for each side. The appropriate pieces are removed or replaced with lower value (or some mechanism to indicate the decrement in capability). There may be several of these engagements each turn and they are done independently of each other. The resulting configuration is the starting point of the next turn.

The combat strength values clearly play a pivotal role in this process. They not only aggregate the combat strengths of the component combat elements, they aggregate this unit's performance over a wide variety of situations. Units that excel in open country maneuver combat might do very poorly in the confined world of street by street, house by house fighting inside cities. Where do these numbers come from? The game designers are usually devoted military historians and military game players who have spent much time designing the rules, combat tables, game piece values so that the outcomes are similar to history. The assignments of combat strengths are based on their assessment of over wide stretches of history of how one nation's soldiers typically measure against others. For a given unit, say a tank battalion, this may be enhanced by their experience in playing more detailed games. However, even with this wealth of background data, the game designers would say that this combat strength is only an indication

of what would happen on a given day. That is why there is the stochastic element in the engagement's combat outcome evaluation.

What are the implications of this for military analysis, in particular, analysis to indicate the benefits of enhancing a weapon system's performance? For a given replication, a particularly propitious stochastic draw at just the right juncture might skew the results. The usual answer to this is to use Monte Carlo simulation – do many replications of the whole process with different stochastic choice in each replication. There are two issues with Monte Carlo techniques – one is that since most of the provided distributions are likely to be normal distributions, the Monte Carlo process will tend to test the centers. The other issue is that if we found that there was a set of results we were particularly interested in (for instance, some replications indicated overall results that were particularly good), we would have difficulty trying to determine which choices might have been that got us there. If we knew, it might very well be valuable to study those parameters more closely.

The Dynamic Focusing Architecture (DFA) was developed to address these issues. It is, in a sense, a controlled Monte Carlo simulator, that instead of using points anywhere in the distribution uses only the centers of mass of partitions of the distribution. For the most part, we partition the distribution into two portions, the portion above the median and the portion below, and we take the average of each, giving us the average low and the average high. In our current engine, we do a replication for each possible way of making a choice leading to 2^n replications, where n is the number of parameters. Even though combinatorial explosive, it provides the testing of the tails and, since for each replication output, we know exactly what choices went into it, we can determine for a given group of outputs, which parameters were primarily responsible. Actually, we can have several parameters returned from each replication, which allows us to do multiple turns. Each of the returned outputs forms a distribution for which we calculate its

average high and average low. These are passed to the next turn as the initial values for that turn. Thus, we can see how a system evolves over multiple time steps.

One caveat is that if there are correlations among these parameters at a given turn (for example, parameter 1 is always inversely proportional to parameter 2 in the outputs), this will be lost by this initialization process. The parameter 1 average high will be chosen 50% of the time whether parameter 2 is high or low.

This example will be a very simple model. We will assume that opposition forces consist of two primary types: core and dispersed. The core elements are distinguished from the dispersed in that they are comparatively highly organized. This organization allows them to perform a wide variety of functions, among which might be superior defense capabilities, but it also makes them vulnerable in that destruction of perhaps a small number of elements could seriously effect the command control necessary to achieve organized action, seriously impairing their capabilities. The dispersed elements are not as instantaneously affected by damage to other dispersed units, but are more vulnerable individually. However, over time, their effectiveness depends on morale – if the core elements start deteriorating or if they hear of other individual dispersed elements being defeated, the fear that they might be left deep in unfriendly territory may start affecting their performance.

```
SLM PROG V2.0
Cell Data;
  RCoreBench : (6, 6);
  RDispBench : (3, 3);
end.

Cell RMorale (10, 10);
// Initial forces
RDisp : (3, 4);
RCore : (6, 8);
SysX : (3.0, 3.1);
SOSC : (7.0, 8.0);
SOSD : (7.0, 8.0);
SysXC : (0.5, 1.0);
SysXD = SysX - SysXC;
BTotC = SysXC + SOSC;
BTotD = SysXD + SOSD;
```

```

// Decr B_on_R_Core
Int1 = BTotC / RCore;
Int1 = 0.15 * Int1;
Int1 = 1.0 - Int1;
// R Core next
Return RCore = RCore * Int1;
// Decr R_on_B_Core
Int1 = RCore / BTotC;
Int1 = 0.1 * Int1;
Int1 = 1.0 - Int1;
// B Core next
BTotCNext = BTotC * Int1;
// Decr B_on_R_Disp
Int1 = BTotD / RDisp;
Int1 = 0.1 * Int1;
Int1 = 1.0 - Int1;
// Effect of R core on R disp
Int2 = RCore / RCoreBench;
Int2 = 0.25 * Int2;
Int2 = 0.75 + Int2;
Int1 = Int2 * Int1;
// R Disp next
Return RDisp = RDisp * Int1;
// Decr R_on_B_Disp
Int1 = RDisp / BTotD;
Int1 = 0.1 * Int1;
Int1 = 1.0 - Int1;
// B Core next
BTotDNext = BTotD * Int1;
// Return RTotForce = R_Core_Next + R_Disp_Next;
// return Blue core values
PercBCdec = BTotCNext / BTotC;
Return SysXC = SysXC * PercBCdec;
Return SOSC = SOSC * PercBCdec;
//return Blue dispersed values
PercBDdec = BTotDNext / BTotD;
SysXD = PercBDdec * SysXD;
Return SysX = SysXD + SysXC;
Return SOSD = PercBDdec * SOSD
// return Red current morale
PercMorale = RDisp / RDispBench;
Return RMorale = PercMorale * RMorale;
end.

```

Figure 3.1: The DFA code for the model

Figure 3.1 contains the DFA code for one of these models. One notices immediately that this is a lengthy, especially for the amount of actual algorithmic content. This is because the DFA engine effectively has to function as a compiler and we haven't fully developed the parser to accept general algebraic expressions – it only accepts unary or binary operations.

The program is divided into cells that can refer to each other using object-based naming techniques. The default is that other cells will take variables from 'Data' if there is no variable of that name within itself. In the cell 'Data' we see two variables, 'RCoreBench' and 'RDispBench'. The pair for each has

duplicated values. This means they are playing the role of constants even though choices will have to be made for them – driving up the computation time. We will discuss why we need to do this later.

The cell 'RMorale' is the one that contains all the computations. It is called 'RMorale' because that parameter is the one we want to see in the graphics. Again, it has the same values for both, but that is because we want it to start from a certain fixed point. Then we have 6 variables mentioned that we will see towards the end as being returned. For these returned values, the next turn will be initialized with the average high and average low of the returned values from this run. Similarly, the next run will be initialized with the returned values from that, etc. RDisp is initialized with average low of 3, average high of 4; RCore is initialized with average low of 6 and average high of 8. For this run, the R side has disposed itself with a heavier concentration on its core forces. For the B side, we have four variables: SysX representing the total combat strength that can be added to B systems, SOSC and SOSD representing the total combat strength of the systems of systems currently addressing core elements and dispersed elements respectively. Finally, SysXC is the amount of strength to be added to the SOSC SoS. The first computation is where the portion of the SysX increment devoted to SOSC, SysXC, is deducted out leaving the remainder that would be devoted to SOSD. We aggregate all the B forces that can be applied to R core and all the B forces that can be applied to R dispersed. We have very simplistic force ratio functions that determine the attrition on both types of R and both types of B forces. For the R dispersed, we have an additional decrement that depends on how the current value of R core compares to the benchmark value, RDispBench, which comes from the 'Data' cell. This is the function of these two variables in the 'Data' cell, to serve as benchmarks for each of the runs, since 'Data' cell variables do not get modified. The attrition to the core and dispersed B systems are reapportioned back to their components (the outputs need to match the inputs). Finally, the R morale is calculated by comparing the R dispersed to its benchmark.

We note that we had started with the R dispersed forces low and the amount of SysX devoted to concentrated as high. Both of these could have other choices, which we have done in other models. Table 3.1 shows the four sets of choices that could be made here. Runs were made for each choice and will be discussed in the following section.

	A	B	C	D
RDisp	(6, 8)	(6, 8)	(3, 4)	(3, 4)
RCore	(3, 4)	(3, 4)	(6, 8)	(6, 8)
SysXC	(0.5, 1.0)	(2.0, 2.5)	(0.5, 1.0)	(2.0, 2.5)
SysXD	(2.0, 2.5)	(0.5, 1.0)	(2.0, 2.5)	(0.5, 1.0)

Table 3.1: Initial values for the four runs

The four runs differ in whether the R side is primarily dispersed (in runs A and B, the dispersed forces are initialized with strengths (6, 8), whereas the core forces are half that) or primarily concentrated (in the other runs, C and D, this allocation of strengths is reversed); and they differ in how B has allocated the incremental improvement for the systems, whether to systems that address the dispersed forces (runs A and C) or that address the core forces (runs B and D). We need to run these as separate runs so that we can carry benchmark values for R's core and dispersed strength that will be used by modules in later turns as a base point. In other words, if the dispersed strength drops below a certain amount of the benchmark, it might signal a significant decision point for the R side.

	A	B	C	D
Turn 1	(7.49, 10.25)	(7.73, 10.44)	(5.68, 8.41)	(6.25, 9.01)
Turn 2	(5.46, 7.91)	(5.72, 8.11)	(2.73, 4.45)	(3.46, 5.79)
Turn 3	(2.80, 4.33)	(2.98, 4.46)	(0.95, 1.77)	(1.55, 2.50)

Table 3.2: results of the 4 runs over three turns

Table 3.2 shows the results for our four runs. Looking at these values for morale, our primary measure in this example, we see very different outcomes for our choices of how which system we increment depending on how the opposition disposes itself. If the opposition is highly dispersed, then putting more into the systems that address the dispersed element gives values of (2.80, 4.33) after three battle turns whereas incrementing systems that are primarily aimed at core type targets gives (2.98, 4.56). Lower values, since this is the opposition's morale, mean better performance. Though somewhat different (about 0.2), this difference is small compared to the overall uncertainty (which is around 1.5). For where the opposition has highly concentrated his forces, we see somewhat different results. For this case, if the opposition is highly dispersed, then putting more into the systems that address the dispersed element gives values of (0.95, 1.77) whereas incrementing systems that are primarily aimed at core type targets gives (1.55, 2.5). In contrast to the first, there is practically no overlap between these two intervals.

This disparity is primarily due to the fact that the morale is calculated from R's dispersed strength (how much it has decreased) and cases C and D have R's strength concentrated in the core elements. Because this example has very simplistic attrition equations, namely that the decrease in strength is proportional to the ratio between the attacker and the defender, the incremental addition to B's anti-dispersed strength gets amplified. In cases A and B, where the dispersed strength is high, the incremental addition does not make as much difference. Further, since the dispersed can also be affected by a decrease in the core strength, an incremental effect on the core can also result in an effect on the dispersed. Thus, we see little difference in what happens to R's dispersed strength due to differences in how B increments the strength of SysXC or SysXD.

This example is 'thought experiment' to show how DFA functions and is not meant to be a valid military combat model. But let's pretend that it is a first cut, high-level model. How do we proceed to improve its fidelity? Our model consists of several computational blocks each

expressing an aspect of how one parameter affects another. Let us look at the block where the total B force that is directed to the R dispersed element does their attrition. The functional description of this would be:

$$\text{DecrementB_on_R_dispersed} (B_{\text{totD}}, R_{\text{Disp}}) = 0.1 (B_{\text{totD}} / R_{\text{Disp}})$$

This is a linear function and, while it may approximate data and experience over some region of the parameters, it clearly is not globally valid. Let us posit that we know the band of values that experience and data provide us and superimpose our initial approximation upon it, which can be seen in Figure 3.2.

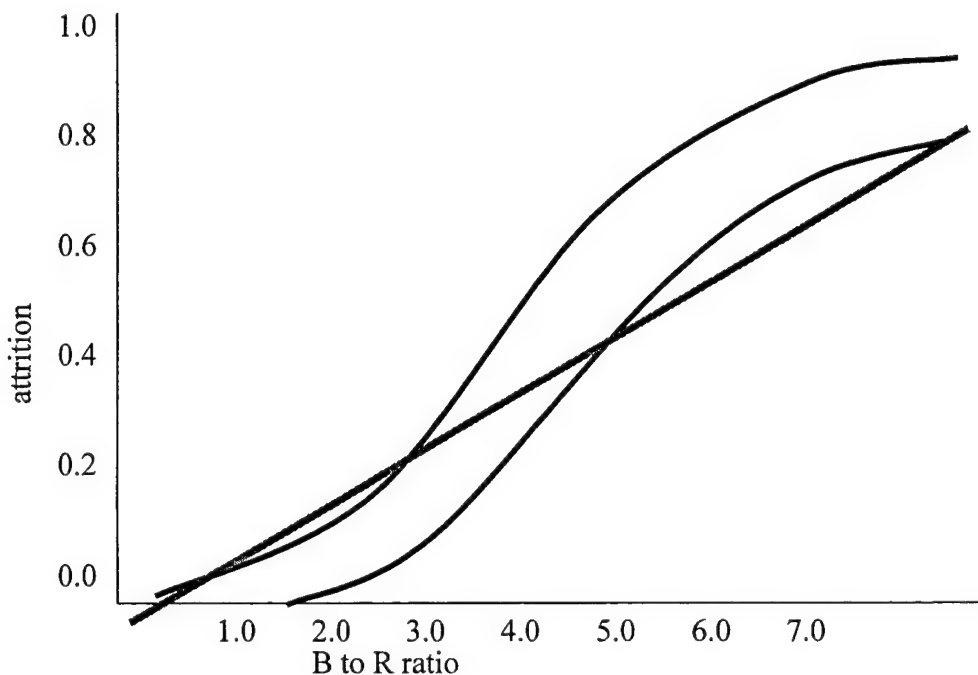


Figure 3.2: Simple linear approximation compared to ‘real’

Our function, $\text{DecrementB_on_R_dispersed} (B_{\text{totD}}, R_{\text{Disp}})$, is shown as the gray line. The two curved lines represent the boundaries of the 50% region, namely 25% is above the top, 25% is below the bottom. Our gray line only sort of approximates this distribution, so we could see that

that might be contributing to our results. But actually, since this overestimates attrition at low ratios (where cases A and B are) and underestimates at higher ratios, the effect we're seeing might actually be understated.

One mechanism to better approximate this distribution is to use a higher order polynomial to better follow the center point of the distribution. Figure 3.3 shows how a curve would better represent the distribution than our original straight line through the origin. We would probably need at least a cubic so as to catch the inflection point, but of course no polynomial will asymptote to 'attrition = 1'. But after some value of the ratio, we could just branch to attrition = 1. The cost of going to a low order polynomial is fairly small computationally – when compared with the costs of paging virtual workspace when we get to large numbers of parameters, it is pretty minimal. Most of the cost is the knowledge engineering to develop and validate the higher fidelity versions.

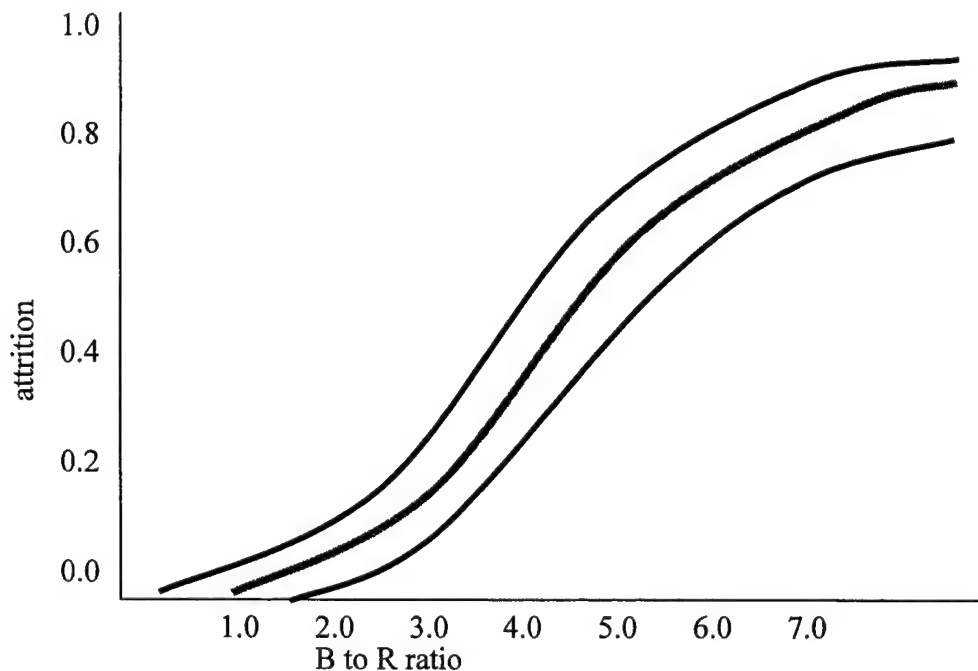


Figure 3.3: A better single valued approximation to the distribution

A better approximation would be one where we capture the breadth of the distribution – that is, we have two lines showing the 50% region, or the standard deviation region or 90% regions. One advantage of taking the average low, average high, is that we can robustly recover the average high, low from almost any computational algorithm, but recovering standard deviations typically requires that we only use addition or only use multiplication (a branching operation would almost certainly lead to unrecoverable standard deviation, at least by moment methods). Another feature of the average high – average low is that the region mapped out is usually quite similar to the 50% region. Figure 3.4 shows these approximations.

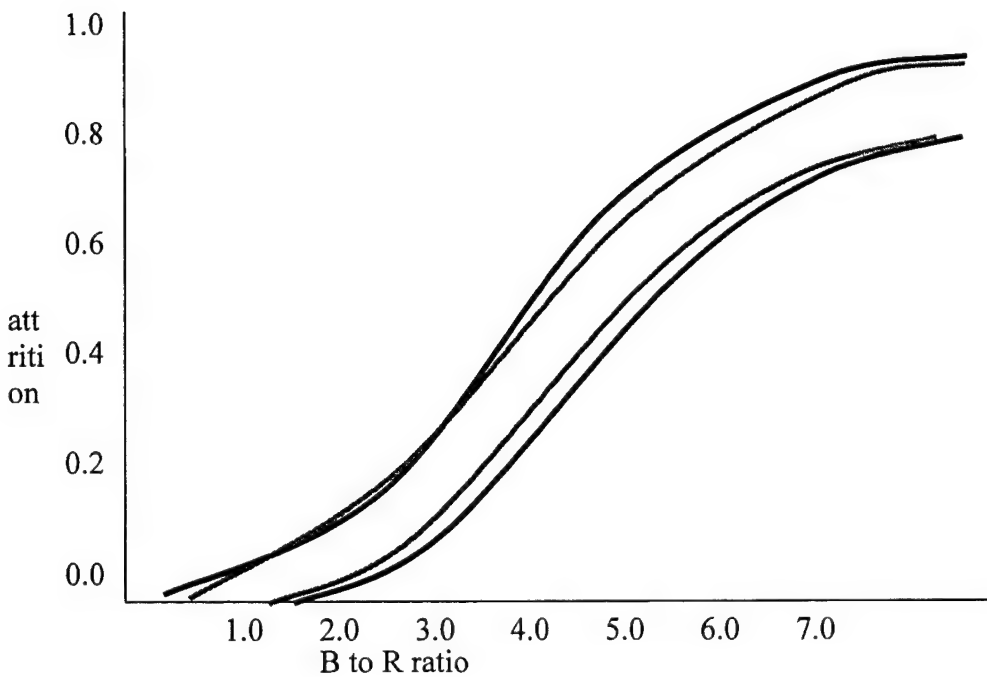


Figure 3.4: Approximating the distribution using the average high, average low

Finally, we can introduce more parameters. Often, when we look at just a few, we have been forced to collapse the knowledge other parameters would provide. For instance, another parameter might be terrain difficulty. For a particular situation, we could move to a more detailed analysis that would yield a distribution substantially tighter than the general case because it takes into account the specifics of this terrain. Figure 3.5 shows how this might considerably reduce the breadth of the distribution – with this narrow band it could very well be that a single valued function, if it stays within this distribution, would be completely adequate for our needs.

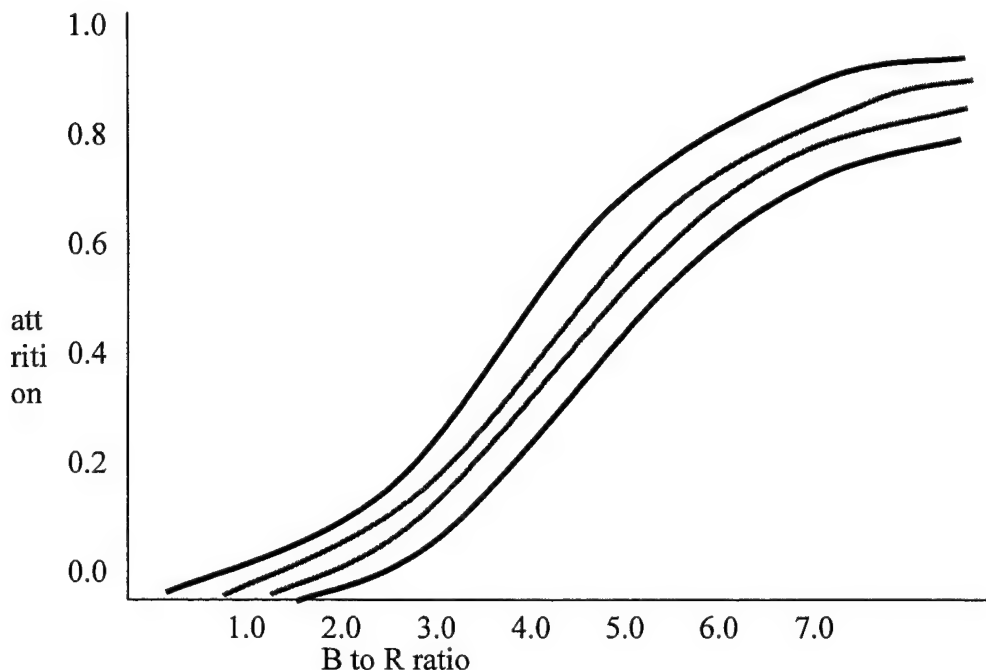


Figure 3.5: Distribution after consideration of other parameters

This process of going to more detail is not computationally expensive. The major issue is that the model structure has been set up to include this refinement as a possible avenue.

The DFA technique is an approach to addressing the issue of ‘stochastic fidelity’, as discussed by C. Cassandrass and W. B. Gong. Their approach is to, instead of partitioning the raw distribution as done here, to first cluster the data into sub-distributions and then use the centroids of each of those as the values that get used in the computations. This clearly produces a very high fidelity simulation, and the DFA could be easily modified to use those instead of the centroids of the high half and the low half. However, since these partitions are not equally weighted, machinery would need to be added to carry the appropriate weights for each cluster. Another feature of DFA is because it is controlled computation, we can recover which parameters were primarily responsible for getting the results into certain regions. Thus, we might start off using the average high – average low for the parameters first and rerun using the clustered values only if that parameter seems to have a significant effect.

3.2 Use of DFA to analyze needs for a military SATCOM system

Though most apparent to the military simulation community, the issue of interrelating models at different levels of aggregation cuts across the whole modeling community. This issue has been a principle focus of AFRL M&S simulation technology development efforts and one of the approaches under their rubric has been the Dynamic Focusing Architecture (DFA). The premise of this approach is that there may not be an automatic method to amalgamate low level results into high level conclusions. The reason for this is that lower level efforts, precisely because they are lower level, will tend to focus on specific things. For instance, in the analysis efforts for Advanced EHF utility, the focus has been on scenarios where there has been a fairly potent land force with significant TBM support attacking our forces (the SWA and NEA scenarios). These fit quite well with the ground rules for this procurement, but down the road, there could be a question “well, just how does this apply to the Kosovo type operations, or Somalia or Sierra Leone...”

This cannot be answered with a single simulator. But, the user community for Advanced EHF would probably feel a lot more comfortable if we had a single analytic process that could address the full range of questions. In the context of our efforts, I am going to try to show how that analytic process could be achieved.

Because of the ground rules of the Advanced EHF procurement, the analysis had been primarily on SWA and NEA scenarios. However, even given the assumptions that underlay those, there could be a substantial amount of disagreement about particulars of these, in particular, contributions of certain elements such as communication. Nichols Research had built a mechanism to address these by having threads that drill down to lower echelons so that force exchange ratios reflect actions at a greater degree of detail. Though this provided a great degree of credibility, nitpickers could still disagree with detail assumptions. For example, in the tracked combat vehicle business a long time ago all our ground vehicle designs were evaluated for performance in simulations that took place in a 40 KM x 40 KM square near Fulda. One marketing person was heard to remark "God forbid we should ever have to fight somewhere else."

In current times, we now know we will almost certainly be fighting somewhere else from wherever the current focus area happens to be. If our analytic process could account for a large number of different assumptions showing results that pertain universally, then we can justifiably assert that we have exercised due diligence in our engineering design. This DFA tool might be able to aid in this.

There would be several steps of the analytic process. The first step is to define a high level computational flow covering the whole engagement. The flow chart for this, Figure 3.6, is shown here.

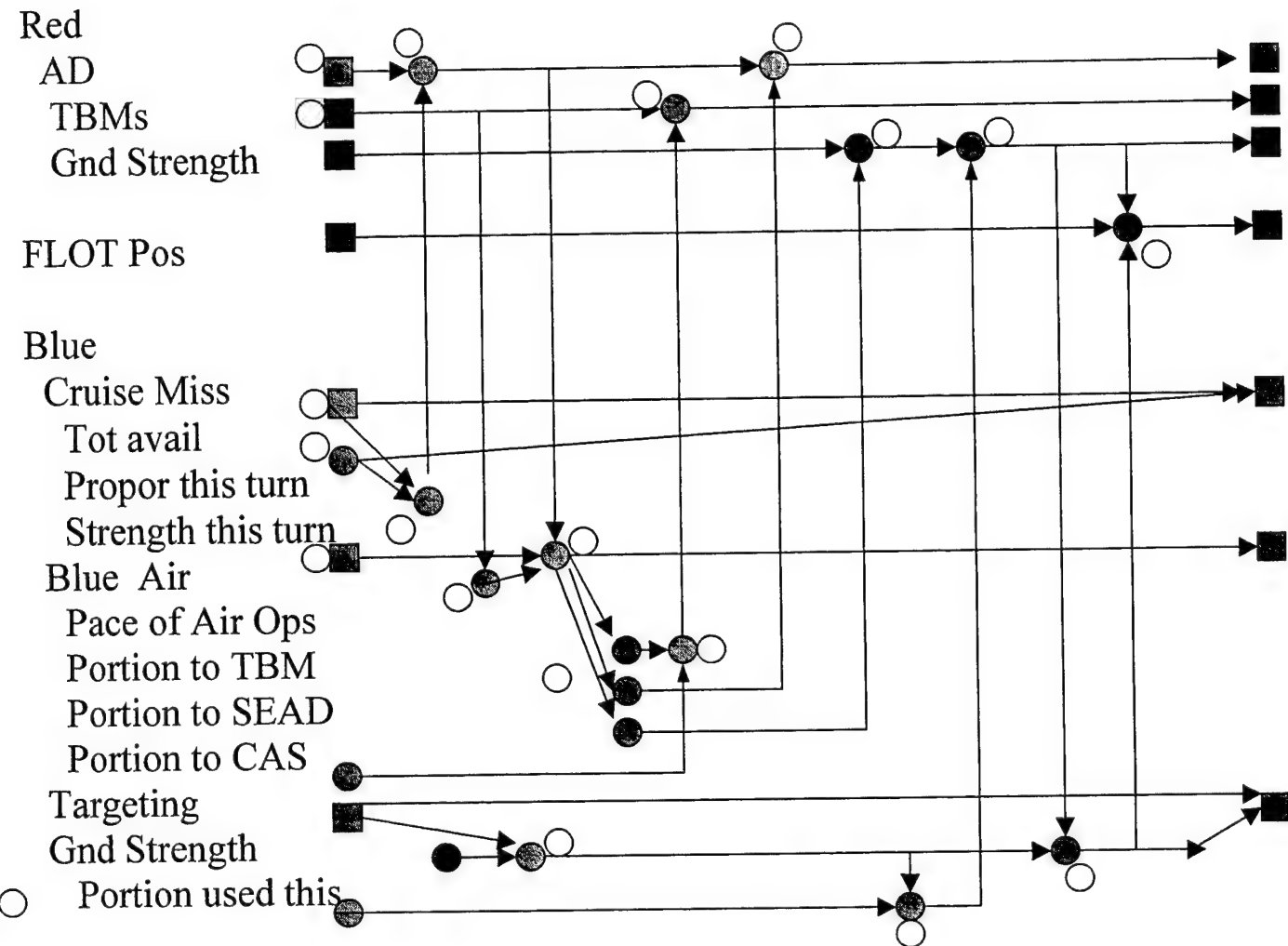


Figure 3.6: Flow Chart One Combat Turn

Each of these lines can be seen as representing a particular variable and the circles represent a computation using all the variables coming into it. Usually this will result in one of the variables being modified, though in a couple of cases it is a new, temporary variable being generated. These computations are then implemented in a driver file for the DFA engine.

The driver file is then executed by the DFA engine. Figure 2 shows a screen dump from the first step of the run of one of the driver files.

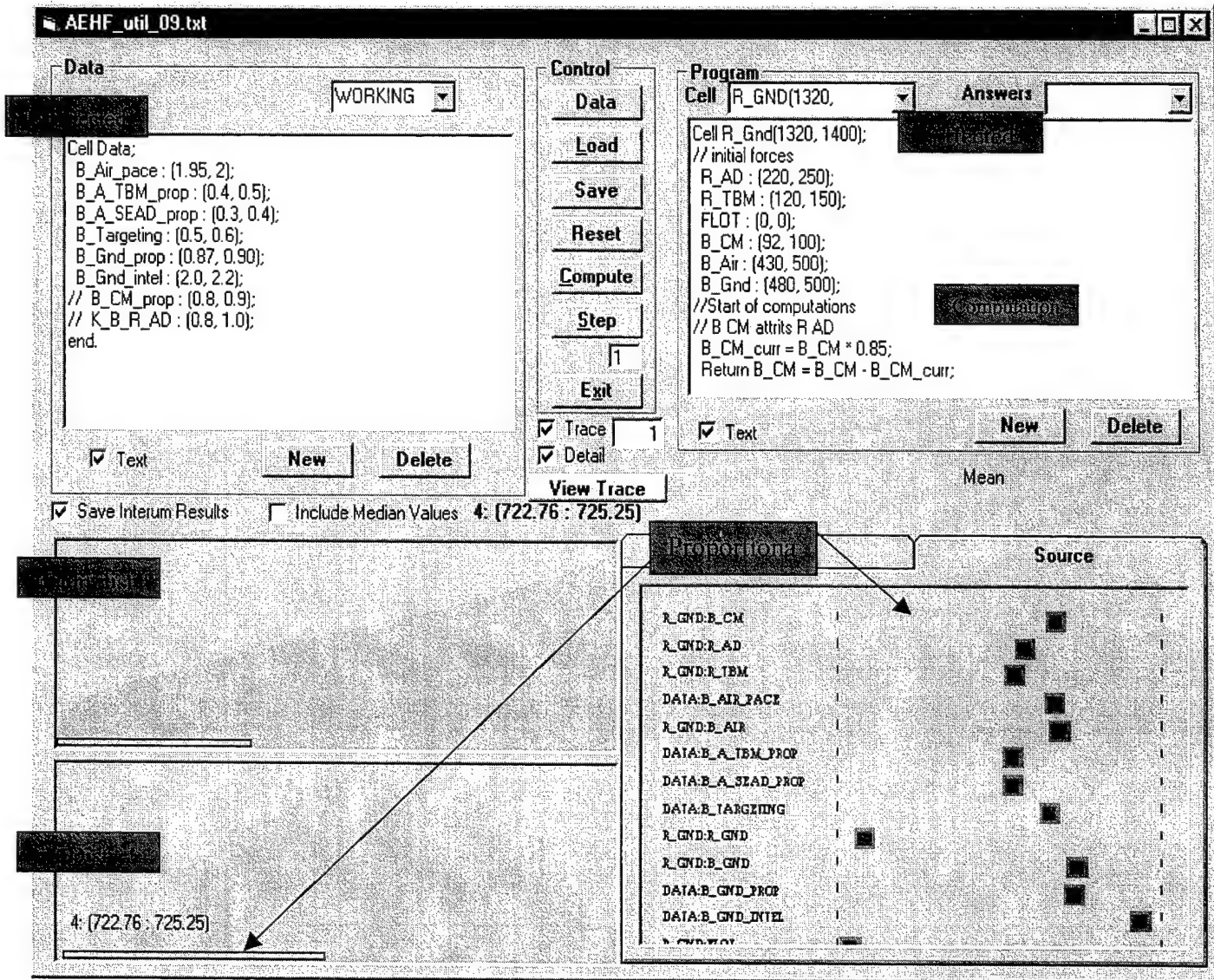


Figure 3.7: A screen dump of our model

Figure 3.7 shows a screen dump of the model. The top portion is devoted to the code and the settings for the model. The left hand side are variables with their two values that will be tested which will be the same pair from combat turn to combat turn. These are primarily proportioning

factors, but we have two communications related variables, B_Targeting and B_Gnd_Intel, that do the force multiplication for their appropriate areas. The right side shows the variables that will change from combat turn to combat turn. The lower left shows the probability distribution of the outputs for Red ground forces (the 2^{13} computations with each possible choice of the two possibilities for each of thirteen variables). The rectangle on the lower left edge of the lower graph shows a portion of the value range, namely a portion where Red has not done well. The lower right graph shows the variables that were important in getting into that region. It shows the proportion of high choices to low choices. From this, we can see that, of course, if we start with a low Red ground force (R_GND:R_GND), we will more than likely end with a low Red ground force – well, actually, it says if we end with a low Red ground force, then we likely started with a low Red ground force. Interestingly, it also says that if we end with a low Red ground force, then we probably had the more effective Blue ground communications (DATA:B_GND_INTEL).

This can be followed through to subsequent combat turns. One hits the ‘step’ key and for the variables that change, we extract an average of the low outputs and an average of the highs and use that to initialize the next turn. This captures the nonlinear feedback behavior that can result in chaotic dynamic systems. More importantly, it allows us to watch how certain choices can cause evolution to vary at a high level. Those variables can then be studied in more detail. Detailed study of the other variables can be postponed, so this provides a mechanism to focus our scarce analysis resources into the most productive avenues.

Now let us add in the ability to do branching. The thrust of the simulation was to start with the distributions describing air and ground forces of the Red and Blue side, apportion them out, match appropriate force elements against each other, have the appropriate attritions happen, then collect together the overall forces and pass those distributions to the next battle turn as starting points. However, without branching, the apportionment process is constant independent of the

values the various values attain and cannot be changed. In particular, the Blue air force apportionment for suppression of enemy air defense (SEAD) could not be changed in response to the amount of air defense (AD) there was, so we got cases where the enemy air defense ended up in the negative region. That is, we had a computational relation for decrementing the Red AD that depended on the Blue Air apportioned to SEAD; so that, decrements to Red AD could be larger than how much Red AD was actually there. With branching, we can test for how much is actually there (for that particular replication) and apportion accordingly.

Thus, branching allows more appropriate planning activities to be simulated. The example in these driver sets, we test for the effect of three different thresholds where we make the branch. Figure 3.8 shows a portion of the code of 'AEHF_util_brnch_05.txt' which contains the branching code. The overall flow of computation was discussed in the earlier report. What we are doing here is to insert a branch operation before the apportioning of the Blue air to SEAD that determines what that portion is. Effectively this says that if the Red AD (R_AD) is less than 100 (and that is in this particular replication), then we will devote a lower proportion of B air to SEAD than otherwise, 0.15 as opposed to 0.35.

```
// Apportion B Air          [e]
  B_A_TBM_str = B_Air_remaining * B_A_TBM_prop;

// branch point
  B_A_SEAD_prop = 0.35;
  IF R_AD < 100 GOTO DOWN;
  GOTO CONTINUE;

DOWN: B_A_SEAD_prop = 0.15;
GOTO CONTINUE;

CONTINUE: B_A_SEAD_str = B_Air_remaining * B_A_SEAD_prop;

  B_A_CAS_int = B_Air_remaining - B_A_SEAD_str;
```



```

B_A_CAS_str = B_A_CAS_int - B_A_TBM_str;
Return R_TBM = R_TBM - B_A_TBM_eff;

```

Figure 3.8: Code for the branching operation

Without the branch, we had to use a distribution of SEAD apportionments, in 'AEHF_util_09.txt', it has values of 0.30 and 0.40, but in any given replication, the choice of which is made 50-50, without respect to the amount of Red AD that needs to be addressed. Two points, one is that this is not very realistic and the other is that adding another choosable variable doubles the computation time. The three drive files 'AEHF_util_brnch_03.txt', '..._05.txt' and '..._04.txt' are exactly the same except for the values for the branch decision, which is 120, 100 and 80, respectively. The extracts of the trace files for these, 'extract branch (03)', etc (Figures 3.9, 3.10 and 3.11 show these values after the third battle turn) which show the average low average high and average of each output variable, show that when we allocate less to SEAD, changing the threshold from 80 to 100 has no effect on the Blue air losses, but the Blue ground forces do better and the Red ground forces do worse. Increasing the threshold to 120 shows now that the Blue air forces are negatively affected, whereas the ground forces do improve. So to go from 100 to 120 would require a trade decision. Going from 80 to 100 has no trade implications, so we could pick 100 because the ground forces do better.

Step:2

```

R_GND:R_GND 74.0665306818328, 191.420490416855 : 128.204528808594
R_GND:B_CM 0.310499846928638, 0.33749982714653 : 0.33749982714653
R_GND:B_AIR 75.2202808866189, 124.868862627419 : 94.642951965332

```

R_GND:R_TBM 6.73377842738174, 80.5787677770246 : 64.1726837158203
R_GND:R_AD 47.0148966317425, 76.6261453624808 : 66.3793411254883
R_GND:B_GND 202.553314254756, 236.389986783636 : 218.665664672852
R_GND:FLOT -36.3951013332228, -26.3662685732211 : -31.5609817504883

Figure 3.9: Output values after third turn for threshold 120

Step:2

R_GND:R_GND 104.295391526526, 211.750005819563 : 157.739944458008
R_GND:B_CM 0.310499846928638, 0.33749982714653 : 0.33749982714653
R_GND:B_AIR 81.130561110983, 143.335557460785 : 109.248466491699
R_GND:R_TBM 2.99313195085147, 78.2861137372692 : 57.2030902771719
R_GND:R_AD 13.032298397265, 76.4232783304503 : 66.1645812988281
R_GND:B_GND 194.247980202656, 225.28329319434 : 209.425918579102
R_GND:FLOT -39.8448200463604, -30.3053914890599 : -35.1555137634277

Figure 3.10: Output values after third turn for threshold 100

Step:2

R_GND:R_GND 123.635702359322, 241.919330315619 : 188.724105834961
R_GND:B_CM 0.310499846928638, 0.33749982714653 : 0.33749982714653
R_GND:B_AIR 81.130561110983, 143.335557460785 : 109.248466491699
R_GND:R_TBM 2.99313195085147, 78.2861137372692 : 57.2030902771719
R_GND:R_AD 13.032298397265, 51.6684613233261 : 27.823299407959
R_GND:B_GND 189.160383969606, 221.324613666172 : 204.249099731445
R_GND:FLOT -42.0027673188511, -31.5853425853092 : -36.7569007873535

Figure 3.11: Output values after third turn for threshold 80

4.0 Issues concerning Metamodeling

The concept of the Metamodel is that it is a distillation of all the analytic activities that pertain to the area so that we can, for a given question, quickly pull out the relevant experiences and how those experiences would be connected (glued together) to provide an assessment. Further, with the DFA tool, we could make determinations of where we most needed to augment the already accomplished efforts, the 'Dynamic Focusing' portion of DFA's name. This could have significant usefulness, for instance in linking together simulators that are of different aggregational levels or were developed with different goals (e.g., cost models and performance models)

4.1 What are metamodels?

When we say 'model', we imply that we have a set of causal relations, usually represented in some computational fashion, that pertains to some situation. This can be problematic since it is expected that the chain of causality can be proved at every step. For a high level model representing some future that is now only dimly perceived, this can be almost impossible, an issue discussed earlier in section 3.2 which discussed traffic modeling for military SATCOM systems. Nobody was willing to sign up for a traffic model, since that would imply a military action in some given theatre, against some given adversary and that the actions of both sides could be predicted in detail. Clearly, nobody would sign up for this, since so much of it is unpredictable; any attempts to foresee a change in operational behavior based on the use of new technologies would collide with the wealth of knowledge about current operational practices.

What we would like to be able to do is to utilize the wealth of knowledge from current operational experience in the very uncertain settings of future scenarios. This means that we need an infrastructure that supports elements of different levels of uncertainty. The metamodel organizes the knowledge, expressing the causalities to the degree of precision that is appropriate. Thus, the burden of proof depends on the degree of uncertainty attached to the causal connection. High certainty requires a high degree of proof, low certainty have commensurately lower standards of proof. This then demands that we have a computational reasoning mechanism that can appropriately combine low certainty causal relations with high certainty ones; the DFA was designed to meet this need.

4.2 How would metamodels work?

The metamodel would be, to the extent possible, the summarization of what is known about a given area including the relations to objects of other metamodels. A typical one would consist of hierarchies of objects, the mathematical expression of the relationship between those objects and the stochastic characterization of parameters of these relations. As discussed above, the variance of these stochastic distributions relates to the degree of proof provided; a narrow distribution requires a high degree of proof and a broad, 'fuzzy' distribution would require less.

An example might be expendable launch vehicles. They have very clear hierarchical decompositions, first into stages, then into components of the stages. There are well-defined computational relations for various measures, such as cost, performance and reliability. For instance, cost is the sum of the costs of the individual elements. But this could include life cycle costs, such as final assembly and test, and the costs of the launch range. Because there are comparatively few launch events, there is not enough history to really put a tight bracket on costs. For that matter, reliability is even more of an issue; currently, if a launch vehicle manufacturer

brings out even a new version of an existing vehicle, he may very well have to fly the first one with a dummy payload. If it is completely new, more than one of these non-revenue launches might be required before the insurers would be willing to back them. The insurers are very conscious of the history of each of the stages, so a modified first stage might be almost as uninsurable as a wholly modified vehicle, but one where the third stage is replaced with another already proven one might be given the same terms.

What we see in this example is that there are examples of decisions being made on levels of uncertainty and even on mixed levels. However, this example is a very restricted decision space; rockets have stages, each stage is definitely independent of the ones that follow and in some senses independent of the preceding ones (the insurance community would not mark a launch failure against a third stage if the failure were traceable to earlier stages). If we wanted to deal in a more general world, we would like to mix levels of uncertainty, but have our end result reflect the impact of the various degrees of uncertainty. To make this happen, we can go to a paradigm where we no longer provide a single number as the result of a study or analysis – rather we present a distribution. If we have done a quick and dirty analysis, then this distribution should probably be very broad. However, if we have a plethora of history or have done a very extensive analysis, then a very narrow distribution would be appropriate.

If we use a stochastic propagation mechanism that keeps an audit trail, we can start off an analysis at a very high level in the hierarchy (low detail, very fuzzy), and be able to backtrack to those objects that are most significant. These can then be opened and examined in more detail. Where there is no information at a more detailed level or what we currently know is too uncertain, more study will be needed; this additional study is now pretty well defined in scope and can probably be done using traditional methodologies.

This allows us to do studies using the extant knowledge and to help focus the requirements for the more traditional analysis methods, thus providing the interface that is required in this more dynamic world between the 'back of the envelope' and the traditional methods.

4.3 Why Metamodeling isn't easy

When this project had been initially conceived several years ago, the thought was that models could be easily integrated along the lines of the Attack Manager Development Facility, a simulation product developed by Rome Laboratories for Strategic Defense Initiative's Directed Energy Office, where outputs from one could serve as inputs for another by a translation function. The DFA, since it is inherently a stochastic process, would extend this idea to models of different aggregational layers. During the course of this effort, the DFA mechanism was extended to be able to have battle turns feed subsequent turns, a crucial computational feature for modeling dynamic behaviors. The DFA was now computationally equipped to play its role, but it had become apparent that this 'gluing' activity required substantial development of knowledge to develop a metamodel.

What has become apparent is that this metamodel cannot just be developed from scratch or even distilled from some collection of component model's documentations. Experience not just in this project, but a number of large communication projects bears this out. We have included an appendix that describes the efforts to delineate a traffic-loading model for Advance EHF. What this points out is that there are whole communities that must buy into a model and that the community has a core set of notions, but many of these date back to the old Inter-German Border scenario set. The ASARC (Army System Acquisition Review Council, a senior management review body) scenario was the central traffic description for the Army, which happened to be the major customer for Adv EHF bandwidth (they accounted for 84% of the bandwidth in the Advanced EHF "Emerging Requirements Database" which was the central traffic description document in early phases of the Advanced EHF procurement). This scenario was the result of

decades of discussion and implementation of mechanisms to provide appropriate communications to the warfighters in the Central Europe (i.e., West Germany) so as to combat an immensely superior numerical force. The difficulty is that with communication systems, the participation of operational elements is key to getting good direction, but communications is to them an ancillary system. So this process of developing a traffic model is protracted and not easy. And, more telling, this traffic model will really only be valid for the past. Trying to keep it current to what is happening now is practically impossible, getting a model that is sufficiently accepted by the community for support to combat operation 10 to 20 years from now is absolutely impossible.

The approach that has been taken on Advanced EHF is to educate the community that we don't necessarily need precise scenarios, rather that we can abstract traffic needs in such a way as to have one abstracted traffic pattern stand for a very wide variety of specific, perhaps very different, detailed scenarios. The issue now is, how do we identify what is the proper abstraction and then how do we sensibly divide that abstracted space. As we pursued this effort, it started to appear that this might be the role DFA could assist. In the next section, we will show that not only does DFA assist analysts with their tasks, as discussed above, but that this activity can be captured and added to the metamodel knowledge base, enhancing other, yet to be conceived, analyses.

4.4 How can we accomplish the metamodel

During the Cold War, it was very clear how to hierarchically decompose the decision space relating to weapon systems: at the toop, National, then Theater, then Echelon Above Corps, then the decompositoin of indiviidual services down through their echelons. Joint activities were coordinated at EAC or above. A given weapon system was 'owned' by an element at certain

levels and the system's role was thus tied to that element's role, which was quite well delineated. In our current situation, and even more so into the future, Joint activities (and even Coalition activities) will be coordinated at lower levels, at least while the activity is happening. This by itself makes the role of a given weapons system less clear, because the role of its owning unit is less clear. Is the traditional hierarchy a sufficient descriptor for crises such as Kosovo? In Section 3.1, we discussed another hierarchy that might have been appropriate for the opposition elements; namely into dispersed and core elements and included a morale factor that linked the two elements to each other and their precious history (in other words, a major strike causing a big decrement of the core forces on one turn would adversely affect the dispersed elements on the next turn). That is, there are adversaries who can attempt to prevail by coordinating forces so that they overcome us at crucial points. For those, organizing them along the traditional hierarchical lines is appropriate. But then there are others who attempt to prevail by just unleashing a horde of personnel and giving them Carte Blanca to do what they feel like. In this case, their targets are not our military elements, but rather some civilian element. For this situation, one effective mechanism is to persuade these dispersed personnel that their core elements might not be able to cover them. They might be left to the wrath of the resident civilians. This lays out a very different dynamic than the traditional military engagement.

New events might cause us to develop new hierarchies, however they could also be the result of further analysis of historical events. Whatever their provenance, these hierarchies will be presented to the community and vetted through the community's process. DFA assists on this side of the metamodel development by infrastructuring the evaluation process and it assists on the other side by guiding new analyses through the collection of hierarchies to those most applicable to the analysis at hand.

5.0 Issues concerning the DFA tool

The Dynamic Focusing Architecture has proven its value in several settings. Though of proven usefulness, it could, like any product, be made better. This tool was aimed at the analysis arena of exploring uncertainty. It has tools that facilitate this exploration, but there are others that could be added. The tools in place provide the analyst with very rapid graphical feedback for a variety of analysis issues: for any of the output variables, a graph of the distributions is presented that is mousable; mousing out any region gives a chart of the percentage of choices made for high and low for each parameter. This provides the analyst with very immediate feedback as to the elements of sensitivity for the problem space being examined.

The major improvement to DFA would be a mechanism that allows the models to be more easily assembled. At present, the mechanism is a file of binary and unary operations, only somewhat higher in accessibility than assembly level code. Being able to use algebraic expressions would be one way of improving accessibility, another would be to provide a 'wiring' paradigm, where outputs from one component could be wired into the inputs of others.

Another issue is to shorten the computation time and the amount of dynamic memory required, since these both limit the number of 'free' variables that can be carried. The first could be addressed by following search strategies that are not exponential. The second, although addressed by this first, could also be addressed by not requiring as much baggage to be carried. This baggage is the graphic knowledge necessary to allow the analyst quick feedback to sensitivity, but this capability can be turned on or off, in the turned off mode, much less dynamic space is required. This both speeds up the computation and allows more free variables to be handled.

6.0 Summary

The modeling and simulation community has been struggling with how to model future events involving large degrees of uncertainty. This is particularly problematic for the warfighters, since they would like to be able to prepare now for future eventualities since so much of their effectiveness involves long lead-time activities. It takes a significant period of time to develop new training procedures and to then actually implement them and train the professionals that will execute. Their tools have a similarly long cycle. New weapon systems need to first have a defined requirement and then there is a lengthy procurement cycle. Modeling can be an important element of validating the requirement, so having a believable model could be a crucial element for anything at all to happen.

The Dynamic Focusing Architecture provides the ability to assess the likely behaviors based on a mix of assumptions, each with different degrees of uncertainty. That is, it can mix different levels of abstraction to provide an appropriate composite view of how these varying degrees of knowledge impact a problem.

This capability may be very important by allowing us to proceed with verifiable analyses even in the very early stages of design and development, the stages typically now done by 'back of the envelope'. As we move into an even less delineated future, this could be a cornerstone capability.

**MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)**

*The advancement and application of Information Systems Science
and Technology to meet Air Force unique requirements for
Information Dominance and its transition to aerospace systems to
meet Air Force needs.*